(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 May 2001 (25.05.2001)

PCT

(10) International Publication Number
WO 01/37067 A1

(51) International Patent Classification[7]: G06F 1/00

(21) International Application Number: PCT/US00/26897

(22) International Filing Date:
29 September 2000 (29.09.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/441,409 16 November 1999 (16.11.1999) US

(71) Applicant *(for all designated States except US)*: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventors; and

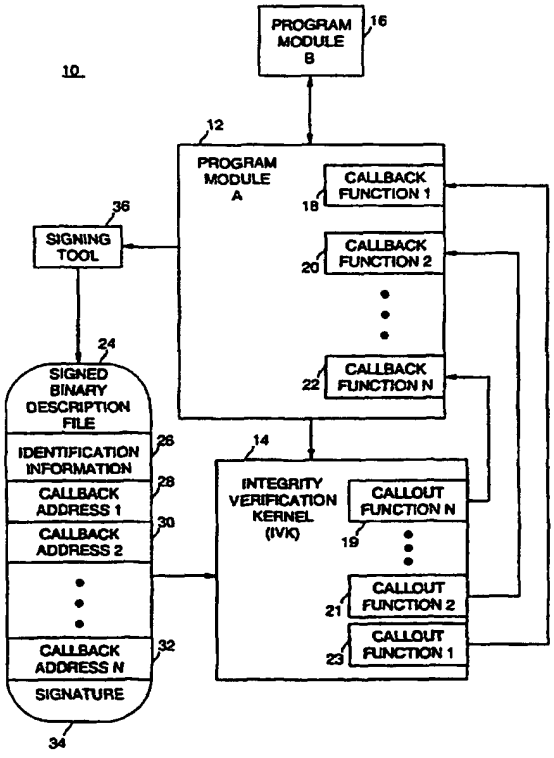
(75) Inventors/Applicants *(for US only)*: **ROTHROCK, Lewis, V.** [US/US]; 17320 NW Woodmere Court, Beaverton, OR 97006 (US). **MALISZEWSKI, Richard, L.** [US/US]; 2218 12th Avenue, Forest Grove, OR 97116 (US). **ROZAS, Carlos, V.** [US/US]; 1534 NW Morgan Lane, Portland, OR 97229 (US). **TUNG, Lihui, C.** [US/US]; 12884 NW Lorraine Drive, Portland, OR 97229 (US). **RANGANATHAN, Kumar** [US/US]; 19000 NW Evergreen Parkway #36, Hillsboro, OR 97124 (US). **PALMER, David, W.** [US/US]; 2014 NW Glisan #211, Portland, OR 97209 (US). **HUDED, Ashok, V.** [US/US]; 15834 NW Rondos Drive, Portland, OR 97229 (US).

(74) Agents: **MALLIE, Michael, J.** et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th Floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR,

[Continued on next page]

(54) Title: A METHOD OF PROVIDING SECURE LINKAGE OF PROGRAM MODULES



(57) Abstract: Secure linkage of first and second program modules so that they may authenticate each other and provide security for digital content accessed by one or more of the modules. The method includes storing at least one address of at least one function of the first program module in a file, calling the second program module by the first program module and passing the file to the second program module, verifying integrity by the second program module of the first program module, and calling, by the second program module, a selected function of the first program module using an address obtained from the file when integrity of the first program module is verified. In one embodiment, the first program module may be a digital content player application and the second program module may be an integrity verification kernel for verifying the integrity of the player application. The file may be a signed binary description file including addresses of functions in the first program module.

WO 01/37067 A1

HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *With international search report.*

— *Before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

A METHOD OF PROVIDING SECURE LINKAGE OF PROGRAM MODULES

BACKGROUND

1. FIELD

The present invention relates generally to digital content protection in computer and consumer electronics systems and, more specifically, to protecting links between program modules.

2. DESCRIPTION

The personal computer (PC) platform is an open and accessible computer architecture. However, the openness of the PC means that it is a fundamentally insecure computing platform. Both the hardware and software can be accessed for observation and modification. This openness allows malicious users and programs to observe and to modify executing code, perhaps with the aid of software tools such as debuggers and system diagnostic tools. Despite these risks, there are classes of operations that must be performed securely on the fundamentally insecure PC platform. These are applications where the basic integrity of the operation must be assumed, or at least verified, to be reliable. Examples of such operations include financial transactions and other electronic commerce, unattended access authorization, and digital content management.

For content providers, countering the threat of digital piracy on the PC requires new software that is resistant to attacks by a malicious user. In this scenario, the malicious user may wish to tamper with or replace particular components of the software in order to gain unauthorized access to digital content or to make unauthorized

reproductions. A cryptosystem based on cryptographic methods employed in conjunction with the software may be used to help protect the content owner's rights. Content may be encrypted to provide some measure of protection, but the software accessing the decrypted content during playback is still vulnerable to attack.

One known approach for protecting such software involves providing a secure linkage between two program modules (e.g., two components of a digital content playback system). This approach includes checking return addresses of function calls from one program module to another and using the addresses to authenticate the identities of the called and calling modules. The secure linkage approach was developed to detect "dead ambassador" attacks where an attacker uses a "dead" or "dummy" module to gain access to a programming interface. In this case the "dummy" module is substituted at run time for the authorized module. Secure linkage may be used for both the calling function and the called function. The concept underlying secure linkage is to capture the return address of the calling function or the address of the called function and verify the addresses against the credentials of an authorized module. A default verification protocol is typically used to determine if an address is within a signed executable portion of the authorized module. More sophisticated protocols verify the address against authorized call points. Thus, secure linkage allows for more secure coupling of modules as well as an access control to internal routines of a module.

Although secure linkage as described above generally provides a known level of security, improvements can still be made to further secure software and stay one step ahead of digital content pirates. What is needed is a method which will allow the fundamentally

insecure, open PC to execute software that is very difficult to be observed or modified.

SUMMARY

An embodiment of the present invention is a method of securely linking first and second program modules so that they may authenticate each other and, in one example, provide security for digital content accessed by one or more of the modules. The method includes storing at least one address of at least one function of the first program module in a file, calling the second program module by the first program module and passing the file to the second program module, verifying integrity by the second program module of the first program module, and calling, by the second program module, a selected function of the first program module using an address obtained from the file when integrity of the first program module is verified.

Other embodiments are described and claimed.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will become apparent from the following detailed description of the present invention in which:

Figure 1 is a diagram of a secure linkage system according to an embodiment of the present invention;

Figure 2 is a flow diagram of secure linkage processing according to an embodiment of the present invention; and

Figure 3 is a diagram illustrating a sample processing system capable of being operated as a secure linkage system according to an embodiment of the present invention.

DETAILED DESCRIPTION

An embodiment of the present invention is a method for redirecting function calls through a protected environment to effect secure linkage of program modules. This method providing secure linkage of two program modules may be accomplished, at least in part, by embedding compiler-specific assembly code into called entry points of a module to capture the caller's return addresses, and in the called exit points to ensure that the proper return addresses are used. An embodiment of the present invention employs a callback methodology to complement return address checks to further protect program module calls. As used herein, a program module is any identifiable portion of computer program code or any sequence of programming instructions in any programming language. Callback functions of a first program module may be registered in credentials for the module. Results from privileged operations of a second program module may be returned using the first module's registered callback functions. Return address checks may be used to determine whether or not to perform the requested operation in a given callback function. Embodiments of the present invention make it more difficult for an attacker to "skip" over the first program module's verification process performed by the second program module, to modify the behavior of the second program module in order to call the first module's services for unauthorized uses, or to replace a legitimate first program module with a rogue module.

Reference in the specification to "one embodiment" or "an embodiment" of the present invention means that a particular feature, structure or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, the

appearances of the phrase "in one embodiment" appearing in various places throughout the specification are not necessarily all referring to the same embodiment.

Figure 1 is a diagram of a secure linkage system 10 according to an embodiment of the present invention. In this embodiment, the concepts of the present invention may be used to securely link a first program module with a second program module. A first program module, program module A 12, uses a second program module, called an Integrity Verification Kernel (IVK) 14, to verify the integrity of the first program module. Embodiments of the present invention operate to make it very difficult to substitute the IVK with another program module (e.g., a hacker's workaround for the verification of the program module) or to observe or modify the internal operation of the IVK. More generally, the two program modules to be securely linked by embodiments of the present invention may comprise any software components. In one embodiment, IVK 14 may be made tamper resistant to further protect itself from attempts by malicious users to tamper with or observe its execution and/or its interaction with program module A. The use of tamper resistance techniques help to provide a protected environment for operation of program module A.

Tamper resistant software is software which has been made resistant to observation and modification. It can be trusted, within certain bounds, to operate as intended even in the presence of a malicious attack on it. Tamper resistant software is position independent and not require relocation in memory. Therefore, tamper resistant software does not have to run in the same address space or processor in which it was created. In one embodiment, such software is generated by using a tamper resistant compiler (not shown). The tamper resistant compiler is a compiler that, when applied to a well prepared software module, replaces the plain-text

source code compiler generated image with a new image that is obfuscated. This self-decrypting software will only execute properly if no part of the image has been altered from the time it was compiled by the tamper resistant compiler. The tamper resistant compiler is a software approach towards providing kernels of software with the ability to run in a "hidden" execution mode. Attempts to decipher what the software is actually doing, or modifications made to the software, will result in the complete failure of the kernels (i.e., it will not decrypt properly).

Program module A 12 may be associated with an Integrity Verification Kernel (IVK) 14. An IVK is software that verifies that the "fingerprint" of an in-memory program image (e.g., the program module A) corresponds to the data within a supplied digital signature. This procedure authenticates the program image (e.g., program module A). It provides a robust mechanism for detecting changes made to executing software, where those changes might be caused by transmission errors or malicious attacks on the software. Any unauthorized change to the software results in a failure in the verification process. If the verification process fails, then other processing related to program module A may be inhibited from executing. For example, if program module A is part of a digital content player application and it's integrity is detected by the IVK to be compromised, then the player application will not decrypt and play the content. IVKs for tamper resistant software are constructed to perform self-checks of object code, bilateral authentication of partner modules, and checks on local and remote data to verify the integrity of a software module. The IVK is self-modifying, self-decrypting, and may be installation unique.

Two intra-process software modules requiring to communicate with each other in an authenticated manner can establish that the module one is calling is indeed the one it is expecting by verifying the digital signature

of the called module using a predetermined "root" key. This process is called bilateral authentication. In embodiments of the present invention, IVK 14 may be used to verify the integrity of program module A 12. The IVK may also be used to perform bilateral authentication between program module A 12 and another program component, such as program module B 16, for example, by using the techniques disclosed in pending U.S. patent application entitled "Method and Apparatus for Integrity Verification, Authentication, and Secure Linkage of Software Modules", Serial No. 09/109,472, which is commonly assigned to the same entity as the present invention. Detailed methods for creating tamper resistant modules and providing integrity verification processing with IVKs and bilateral authentication are described in U.S. patent 5,892,899, entitled "Tamper Resistant Methods and Apparatus" and pending patent application entitled "Tamper Resistant Methods and Apparatus", Serial No. 08/924,740, both of which are commonly assigned to the same entity as the present invention.

According to embodiments of the present invention, program module A 12 comprises one or more callback functions, designated callback function 1 18, callback function 2 20, ... callback function N 22, in Figure 1. A callback function as used herein is a portion of software within program module A that is callable by another program module, such as IVK 14. A callback function may assist in performing a verification operation for program module A. A callback function starts at a callback address within program module A. Each callback function may correspond to a particular program verification or critical section called a "milestone" herein. In at least one embodiment of the present invention there are at least three milestones. One milestone is a verification that program module A has not been tampered with. Another milestone is a verification that no program debuggers are currently being executed by the computer system which is

executing program module A. Such debuggers may be used by malicious users to interfere with, observe, or modify the execution of program module A. Another milestone is a verification that a signed binary description file (BDF) 24 associated with the program module A and IVK combination has been created by a trusted entity.

Signed BDF 24 is a file comprising, at least in part, a binary description of the integrity of program module A. The signed BDF may be stored anywhere that is accessible to program module A and the IVK. In one embodiment, the BDF comprises identification information 26 associating the file with program module A 12 and IVK 14, one or more callback addresses corresponding to callback functions 18, 20, and 22 (denoted callback address 1 28, callback address 2 30, ... callback address N 32, in Figure 1), and a digital signature 34. The signed BDF may be created by a signing tool 36, which accepts information about the callback functions from program module A, and identification information from program module A and the IVK, and signs the file. One suitable signing tool for this purpose is described in pending U.S. patent application entitled "Method and Apparatus for Integrity Verification, Authentication, and Secure Linkage of Software Modules", Serial No. 09/109,472, which is commonly assigned to the same entity as the present invention. In one embodiment, a hash value of program module A may be computed as part of the identification information and stored in the BDF along with the callback addresses. This information may be signed using the private half of an asymmetric key pair.

By using encrypted callback addresses in the signed BDF, it becomes difficult for a hacker to substitute a rogue module for the IVK due to the fact that it becomes very hard to know what address needs to be called for a given callback function.

Program module A calls the IVK to verify the integrity of program module A and passes the associated signed BDF to the IVK. Since the IVK is tamper resistant, it may be difficult for a hacker to observe or modify the IVK. Since the callback addresses are stored in the signed BDF, they cannot be changed without invalidating the cryptographic digital signature stored therein. Using callback addresses according to embodiments of the present invention adds extra security to this arrangement of modules by providing a level of indirection in the secure linkage of the modules. This scheme ties the delivery of program module A's benefit to an integrity check by the IVK so that a user may not get the benefit without performing the integrity check. Further, the IVK cannot easily be replaced by a rogue module because of the encrypted callback addresses. It is thus more difficult to separate or override the linkage between program module A and the IVK.

The verification operation may be divided into one or more milestones. When each milestone is completed successfully, a callback address associated with the milestone may be retrieved from the signed BDF, relocated, and verified to be a valid address in program module A. If the callback address is valid, the associated callback function may be called or otherwise operated on. The callback function performs some critical work required for the delivery of services by program module A (such as playing digital content, for example), and may return an error indication in case of failure. In one embodiment, no error indication is returned; the program module may simply stop executing. In this manner, the verification of program module A is required for successful operation of critical functions of the module. Therefore, it becomes difficult for an attacker to skip over the call to the IVK.

Milestone callout functions (denoted callout function 1 19, callout function 2 21, ... callout function N 23) in the IVK corresponding to the

callback functions may be defined to provide secure linking verification of program module A with delivery of critical program services. Milestone callout functions comprise portions of software within the IVK for implementing verification actions. These milestone callout functions may be obfuscated to make them tamper resistant when the IVK is built. Each milestone callout function may call an associated callback function in program module A.

Figure 2 is a flow diagram of secure linkage processing according to an embodiment of the present invention. At block 100, an operating system (OS) loads and begins execution of program module A 12. At block 102, program module A passes the signed binary description file (BDF) to the IVK 14 to verify the module's integrity before continuing processing. Thus, program module A calls the IVK with the signed BDF as an input parameter. Next, at block 104, the IVK performs verification of program module A by utilizing the signed BDF for at least one milestone. For example, a milestone may be checking the signature in the BDF against a computed value for program module A. If the values do not match, then the program module must have been tampered with. If the milestone verification is not a success at block 106, the IVK stops the verification process for program module A and returns an indication of failure to program module A at block 108. Program module A then knows that it has been compromised and may abort without performing any further processing.

If the current milestone verification is a success at block 106, then verification processing continues with block 110. At this block, the IVK obtains the callback address associated with the current milestone from the signed BDF, relocates it, verifies that the callback address is within program module A, and performs an operation using the callback function. That is, the IVK calls the callback function addressed by the current

callback address within program module A. In one embodiment, a milestone callout function (e.g., callout function 1 19) within the IVK calls a callback function in program module A. The callback function executes to perform some useful function for program module A. Generally, this operation may be critical for the successful operation of program module A. For example, this could include a partial decryption of digital content. When the callback function finishes, it returns control back to the callout function that called it within the IVK. If all milestones are not complete at block 112, the IVK checks the next milestone at block 104. If all milestones are complete at block 112, then IVK completes verification of program module A at block 114 and returns an indication of successful verification to program module A. At this point, control may be passed back to program module A for continued processing.

The present invention comprises a general mechanism for linking tamper resistant code with other tamper resistant code or plain text code in such a way as to deter tampering or other unauthorized access. Different embodiments of the present invention may comprise various options to the above described technique. For example, the secure linkage processing may be applied to modules other than IVKs. Milestones may be repeatedly checked to provide a continual self-check of program module A while it is running, thereby detecting any attempts to tamper with the module while it is running. The tamper resistance may be implemented in hardware, rather than in software.

Another embodiment of the present invention strengthens the security further by encrypting the callback addresses in the signed BDF and decrypting them prior to use by the IVK. The cryptographic key may be hidden in such a way as to be available for the IVK when needed. If the callback addresses or callback functions have been tampered with, an error indication may be generated when the callback address is called or

referenced. In another embodiment, the callback functions may be encrypted and decrypted just prior to use. To add further protection, a milestone callout function may be invoked in the context of an exception handler within the IVK that will trap invalid operations. In this way, the IVK maintains control over failure cases and it becomes even more difficult for an attacker to detect the failed callback. Thus, the attacker merely observes that the program module doesn't work, but cannot determine why it does not work.

In other embodiments, additional variations of secure linkage may be implemented. Customized milestone callout functions may invoke corresponding callout functions with an arbitrary number of parameters. The customized milestone callout functions may themselves be tamper resistant code. The callback address passed into a customized milestone callout function may be used as a pointer into a data store or to operate on a data structure, perhaps using secret information stored in the tamper resistant code of the IVK to provide a further level of indirection. The callback address passed into a milestone callout function may be used as a pointer to a data store holding object code for a "missing function" which may be moved into the "correct" place in the IVK, effectively enabling the program module to work correctly only after the program module has been verified. The callback function's return value may be used as a function pointer. In this way, callbacks may be chained together, thereby allowing a series of functions to be called. The specific callback could also depend on the caller's address, causing different program behavior depending on the specific location of the calling function.

In the preceding description, various aspects of the present invention have been described. For purposes of explanation, specific numbers, systems and configurations were set forth in order to provide a

thorough understanding of the present invention. However, it is apparent to one skilled in the art having the benefit of this disclosure that the present invention may be practiced without the specific details. In other instances, well-known features were omitted or simplified in order not to obscure the present invention.

Embodiments of the present invention may be implemented in hardware or software, or a combination of both. However, embodiments of the invention may be implemented as computer programs executing on programmable systems comprising at least one processor, a data storage system (including volatile and non-volatile memory and/or storage elements), at least one input device, and at least one output device. Program code may be applied to input data to perform the functions described herein and generate output information. The output information may be applied to one or more output devices, in known fashion. For purposes of this application, a processing system embodying the playback device components includes any system that has a processor, such as, for example, a digital signal processor (DSP), a microcontroller, an application specific integrated circuit (ASIC), or a microprocessor.

The programs may be implemented in a high level procedural or object oriented programming language to communicate with a processing system. The programs may also be implemented in assembly or machine language, if desired. In fact, the invention is not limited in scope to any particular programming language. In any case, the language may be a compiled or interpreted language.

The programs may be stored on a storage media or device (e.g., hard disk drive, floppy disk drive, read only memory (ROM), CD-ROM device, flash memory device, digital versatile disk (DVD), or other storage device) readable by a general or special purpose programmable processing system, for configuring and operating the processing system

when the storage media or device is read by the processing system to perform the procedures described herein. Embodiments of the invention may also be considered to be implemented as a machine-readable storage medium, configured for use with a processing system, where the storage medium so configured causes the processing system to operate in a specific and predefined manner to perform the functions described herein.

An example of one such type of processing system is shown in Figure 3, however, other systems may also be used and not all components of the system shown are required for the present invention. Sample system 400 may be used, for example, to execute the processing for embodiments of the secure linkage system, in accordance with the present invention, such as the embodiment described herein. Sample system 400 is representative of processing systems based on the PENTIUM®II, PENTIUM® III, and CELERON™ microprocessors available from Intel Corporation, although other systems (including personal computers (PCs) having other microprocessors, engineering workstations, other set-top boxes, and the like) and architectures may also be used.

Figure 3 is a block diagram of a system 400 of one embodiment of the present invention. The system 400 includes a processor 402 that processes data signals. Processor 402 may be coupled to a processor bus 404 that transmits data signals between processor 402 and other components in the system 400.

System 400 includes a memory 406. Memory 406 may store instructions and/or data represented by data signals that may be executed by processor 402. The instructions and/or data may comprise code for performing any and/or all of the techniques of the present invention. Memory 406 may also contain additional software and/or data (not shown). A cache memory 408 may reside inside processor 402 that stores data signals stored in memory 406.

A bridge/memory controller 410 may be coupled to the processor bus 404 and memory 406. The bridge/memory controller 410 directs data signals between processor 402, memory 406, and other components in the system 400 and bridges the data signals between processor bus 404, memory 406, and a first input/output (I/O) bus 412. In this embodiment, graphics controller 413 interfaces to a display device (not shown) for displaying images rendered or otherwise processed by the graphics controller 413 to a user.

First I/O bus 412 may comprise a single bus or a combination of multiple buses. First I/O bus 412 provides communication links between components in system 400. A network controller 414 may be coupled to the first I/O bus 412. In some embodiments, a display device controller 416 may be coupled to the first I/O bus 412. The display device controller 416 allows coupling of a display device to system 400 and acts as an interface between a display device (not shown) and the system. The display device receives data signals from processor 402 through display device controller 416 and displays information contained in the data signals to a user of system 400.

A second I/O bus 420 may comprise a single bus or a combination of multiple buses. The second I/O bus 420 provides communication links between components in system 400. A data storage device 422 may be coupled to the second I/O bus 420. A keyboard interface 424 may be coupled to the second I/O bus 420. A user input interface 425 may be coupled to the second I/O bus 420. The user input interface may be coupled to a user input device, such as a remote control, mouse, joystick, or trackball, for example, to provide input data to the computer system. A bus bridge 428 couples first I/O bridge 412 to second I/O bridge 420.

Embodiments of the present invention are related to the use of the system 400 as a secure linkage system. According to one embodiment,

such processing may be p rformed by the system 400 in response to processor 402 executing sequences of instructions in memory 404. Such instructions may be read into memory 404 from another computer-readable medium, such as data storage device 422, or from another source via the network controller 414, for example. Execution of the sequences of instructions causes processor 402 to execute secure linkage processing according to embodiments of the present invention. In an alternative embodiment, hardware circuitry may be used in place of or in combination with software instructions to implement embodiments of the present invention. Thus, the present invention is not limited to any specific combination of hardware circuitry and software.

The elements of system 400 perform their conventional functions in a manner well-known in the art. In particular, data storage device 422 may be used to provide long-term storage for the executable instructions and data structures for embodiments of the secure linkage system in accordance with the present invention, whereas memory 406 is used to store on a shorter term basis the executable instructions of embodiments of the secure linkage system in accordance with the present invention during execution by processor 402.

While this invention has been described with reference to illustrative embodiments, this description is not intended to be construed in a limiting sense. Various modifications of the illustrative embodiments, as well as other embodiments of the invention, which are apparent to persons skilled in the art to which the inventions pertains are deemed to lie within the spirit and scope of the invention.

CLAIMS

What is claimed is:

1. A method of securely linking first and second program modules comprising:

storing at least one address of at least one function of the first program module in a file;

calling the second program module by the first program module and passing the file to the second program module;

verifying integrity, by the second program module, of the first program module; and

calling, by the second program module, a selected function of the first program module using an address obtained from the file when integrity of the first program module is verified.

2. The method of claim 1, wherein the file comprises at least one of a digital signature of the at least one address and a digital signature of the first program module.

3. The method of claim 2, wherein verifying integrity comprises at least one of validating the digital signature of the at least one address and validating the digital signature of the first program module.

4. The method of claim 1, wherein the file comprises identification information associating the first program module with the second program module.

5. The method of claim 1, wherein the second program module comprises tamper resistant software.

6. The method of claim 1, wherein verifying integrity comprises bilateral authentication of the first and second program modules.

7. The method of claim 1, wherein verifying integrity comprises at least one of verifying that the first program module has not been modified, determining that no program debugger applications are executing on a processing system executing the program modules, and verifying that the file has been created by a trusted entity.

8. The method of claim 1, wherein verifying integrity comprises validating that the address is a calling address of the selected function within the first program module.

9. The method of claim 1, wherein storing the at least one address in the file further comprises encrypting the at least one address during storage.

10. The method of claim 1, wherein the at least one function comprises tamper resistant software.

11. The method of claim 1, wherein calling the selected function comprises calling the selected function with an arbitrary number of parameters.

12. An article comprising: a storage medium having a plurality of machine readable instructions, wherein when the instructions are executed

by a processor, the instructions provide for the secure linkage of first and second program modules by

storing at least one address of at least one function of the first program module in a file;

calling the second program module by the first program module and passing the file to the second program module;

verifying integrity, by the second program module, of the first program module; and

calling, by the second program module, a selected function of the first program module using an address obtained from the file when integrity of the first program module is verified.

13. The article of claim 12, wherein the file comprises at least one of a digital signature of the at least one address and a digital signature of the first program module.

14. The article of claim 13, wherein instructions for verifying integrity comprises instructions for at least one of validating the digital signature of the at least one address and validating the digital signature of the first program module.

15. The article of claim 12, wherein the file comprises identification information associating the first program module with the second program module.

16. The article of claim 12, wherein the second program module comprises tamper resistant software.

17. The article of claim 12, wherein instructions for verifying integrity comprises instructions for bilateral authentication of the first and second program modules.

18. The article of claim 12, wherein instructions for verifying integrity comprises instructions for at least one of verifying that the first program module has not been modified, for determining that no program debugger applications are executing on a processing system executing the program modules, and for verifying that the file has been created by a trusted entity.

19. The article of claim 12, wherein instructions for verifying integrity comprises instructions for validating that the address is a calling address of the selected function within the first program module.

20. The article of claim 12, wherein instructions for storing the at least one address in the file further comprise instructions for encrypting the at least one address during storage.

21. The article of claim 12, wherein the at least one function comprises tamper resistant software.

22. The article of claim 12, wherein instructions for calling the selected function comprise instructions for calling the selected function with an arbitrary number of parameters.

23. A method of providing integrity verification of a first program module by a second program module, the first program module having addresses of callable functions being stored in a file, comprising:

calling the second program module by the first program module to verify integrity of the first program module;

performing integrity verification of the first program module by the second program module using the file for at least one verification milestone; and

obtaining a selected address from the file, verifying that the address is within the first program module, and calling a selected function addressed by the selected address and associated with the at least one verification milestone when the verification milestone is successfully reached.

24. The method of claim 23, wherein the at least one verification milestone comprises at least one of verifying that the first program module has not been modified, determining that no program debugger applications are executing on a processing system executing the program modules, and verifying that the file has been created by a trusted entity.

25. The method of claim 23, wherein the file comprises a digital signature of the addresses.

26. The method of claim 23, further comprising calling the second module, performing integrity verification, obtaining the selected address, verifying the selected address, and calling the first program module repeatedly throughout execution of the first program module to detect any compromise of the integrity of the first program module.

27. The method of claim 23, wherein the first program module comprises a digital content player application and the second program module comprises an integrity verification kernel.

28. The method of claim 23, wherein the addresses are encrypted in the file and obtaining the selected address comprises decrypting the selected address.

29. The method of claim 23, wherein the second program module is tamper resistant.

30. An article of manufacture comprising: a storage medium having a plurality of machine readable instructions, wherein when the instructions are executed by a processor, the instructions provide for integrity verification of a first program module by a second program module, addresses of callable functions of the first program module being stored in a file, the instructions for

calling the second program module by the first program module to verify integrity of the first program module;

performing integrity verification of the first program module by the second program module using the file for at least one verification milestone; and

obtaining a selected address from the file, verifying that the address is within the first program module, and calling a selected function addressed by the selected address and associated with the at least one verification milestone when the verification milestone is successfully reached.

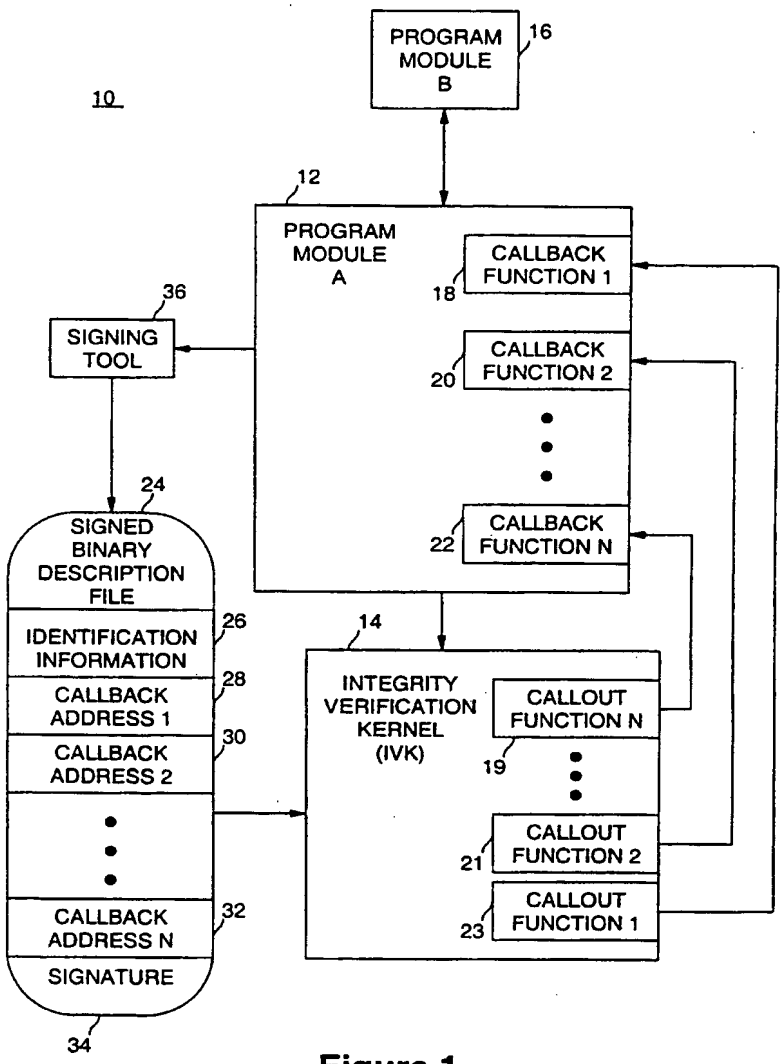

10
PROGRAM MODULE B
16
12
PROGRAM MODULE A
CALLBACK FUNCTION 1
18
CALLBACK FUNCTION 2
20
CALLBACK FUNCTION N
22
36
SIGNING TOOL
24
SIGNED BINARY DESCRIPTION FILE
IDENTIFICATION INFORMATION
26
CALLBACK ADDRESS 1
28
CALLBACK ADDRESS 2
30
CALLBACK ADDRESS N
32
SIGNATURE
34
14
INTEGRITY VERIFICATION KERNEL (IVK)
CALLOUT FUNCTION N
19
CALLOUT FUNCTION 2
21
CALLOUT FUNCTION 1
23

Figure 1

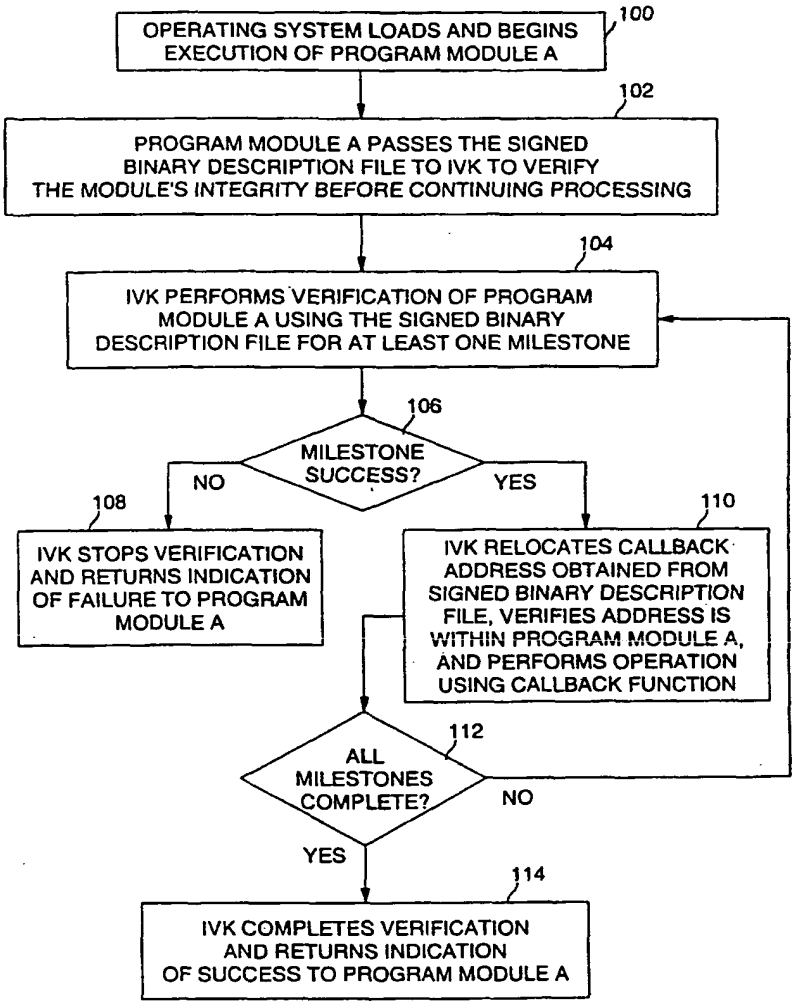

OPERATING SYSTEM LOADS AND BEGINS EXECUTION OF PROGRAM MODULE A
100
PROGRAM MODULE A PASSES THE SIGNED BINARY DESCRIPTION FILE TO IVK TO VERIFY THE MODULE'S INTEGRITY BEFORE CONTINUING PROCESSING
102
IVK PERFORMS VERIFICATION OF PROGRAM MODULE A USING THE SIGNED BINARY DESCRIPTION FILE FOR AT LEAST ONE MILESTONE
104
MILESTONE SUCCESS?
106
NO
YES
IVK STOPS VERIFICATION AND RETURNS INDICATION OF FAILURE TO PROGRAM MODULE A
108
IVK RELOCATES CALLBACK ADDRESS OBTAINED FROM SIGNED BINARY DESCRIPTION FILE, VERIFIES ADDRESS IS WITHIN PROGRAM MODULE A, AND PERFORMS OPERATION USING CALLBACK FUNCTION
110
ALL MILESTONES COMPLETE?
112
NO
YES
IVK COMPLETES VERIFICATION AND RETURNS INDICATION OF SUCCESS TO PROGRAM MODULE A
114

Figure 2

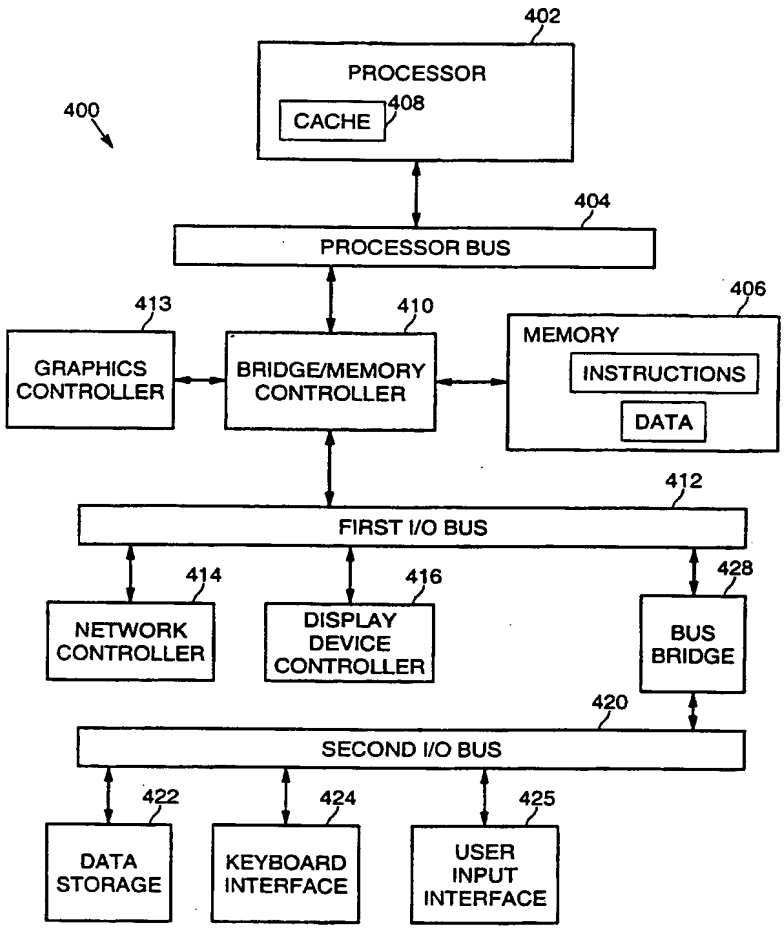

400
402
PROCESSOR
CACHE
408
404
PROCESSOR BUS
406
413
410
GRAPHICS CONTROLLER
BRIDGE/MEMORY CONTROLLER
MEMORY
INSTRUCTIONS
DATA
412
FIRST I/O BUS
414
416
428
NETWORK CONTROLLER
DISPLAY DEVICE CONTROLLER
BUS BRIDGE
420
SECOND I/O BUS
422
424
425
DATA STORAGE
KEYBOARD INTERFACE
USER INPUT INTERFACE

Figure 3

INTERNATIONAL SEARCH REPORT

In ational Application No
PCT/US 00/26897

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F1/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 689 565 A (SPELMAN JEFFREY F ET AL) 18 November 1997 (1997-11-18) column 3, line 5 -column 4, line 8	1-30
A	EP 0 845 733 A (SUN MICROSYSTEMS INC) 3 June 1998 (1998-06-03) column 3, line 35 -column 4, line 28	1-30
A	EP 0 770 957 A (SUN MICROSYSTEMS INC) 2 May 1997 (1997-05-02) page 3, line 50 -page 5, line 38	1-30

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
9 March 2001	20/03/2001

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Bijn, K

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Int tional Application No
PCT/US 00/26897

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5689565 A	18-11-1997	NONE	
EP 0845733 A	03-06-1998	US 6021491 A	01-02-2000
		US 5958051 A	28-09-1999
		JP 10326078 A	08-12-1998
EP 0770957 A	02-05-1997	US 5757914 A	26-05-1998
		JP 9231068 A	05-09-1997
		TW 378304 B	01-01-2000
		US 5970145 A	19-10-1999